In the Specification

Please amend the specification of this application as follows:

Rewrite Table 1 at page 29 as follows:

--

Development Process	Information Type			
	Program Flow/ Timing	Memory Reads and Writes	ASIC Data/ Other Activity	Application Data
Debug				
Run away code	X			
Memory corruption	X	X		
General trace	X	X		X
ASIC/peripherals operation			X	X
Input and/or output data			X	X
Quality Assurance				
Code coverage	X			
Path coverage	X			
Optimization				
Program profiling	X			
Bus or memory profiling	X			

Table 1--

Rewrite the paragraph at page 30, line 20 to page 31, line 2 as follows:

--After compression, the data is exported at a programmable or fixed trace port width at a clock rate that may or may not be the same as the CPU core 201 clock rate. The export clock is derived from either CPU core 201 clock or from a local oscillator dedicated to the generation of the export clock. The local oscillator option addresses several issues. As system clock rates increase, it may not be possible to export data at CPU core 201 clock rate. An export clock not derived from CPU clock 201 rate may maximize the

export bandwidth. Using an optional local oscillator incurs a small incremental hardware cost in order to maximize the transmission bandwidth of each trace pin.--

Rewrite the paragraph at page 31, line 28 to page 32, line 10 as follows:

--Advanced analysis facilities or components identify what data is to be collected or how much data is to be collected. These facilities are used to start or stop trace acquisition after a sequence of events occurs in the application program or directly select the data that is placed in collection FIFOs. Advanced analysis also provides the ability to collect a pre-specified number of bus operations or transactions after the occurrence of an event or define an event that ends the trace session collecting N trace transactions before the system event. These facilities can be used to create a number of additional filtering criteria, with the richness of this criteria set by the strength of the analysis event ~~detection,~~ detection state machines and counters.--

Rewrite the paragraph at page 32, lines 11 to 26 as follows:

--Debugger application program 110 can be used to collect system activity when the data of interest can be observed at a point in the applications program. In this case, code is added to the application to collect the desired information and move this information to the trace logic for export. This is approach is actually a hybrid of real-time data exchange and trace where CPU core 201 collects the data which the trace export mechanism exports. This collection and export mechanism provides a high bandwidth output-only application accessible port. This capability can be used simultaneously and in conjunction with the bus snooper capabilities. The applications program collection method targets data collection and is not suited for the collection of program

flow and timing information. It is more cost effective than the bus snooper method when used to collect data streams that are related to program flow.--

Rewrite the paragraph at page 39, line 22 to page 40, line 5 as follows:

--Figure 4 illustrates the final trace output stages. Trace export 240 includes trace packet build 242, first-in-first-out (FIFO) buffer 246 and transmission formatter 248. Pin manager and pin macros 260 includes pin manager 261 and pin macros 262. As illustrated in Figure 4, trace packet ~~built~~ build 242 and the input stage to FIFO buffer 246 are clocked by CPU_CLK, which is employed by CPU core 201. The output stage of FIFO buffer 246 as well as transmission formatter 248, pin manager 261 and pin macros 262 are optionally clocked by either CPU_CLK or by a different clock signal from transmission clock ~~generator~~ generation 245 (called local oscillator 245 in Figure 3). Transmission clock ~~generator~~ generation 245 is generic to all instruction set architectures (ISAs) and ~~reused~~ reuse of the implementation is expected.--

Rewrite the paragraph at page 40, lines 6 to 25 as follows:

--FIFO buffer 246 is either: asynchronous to CPU_CLK clock; synchronous to CPU_CLK; or not needed. When FIFO buffer 246 is asynchronous to CPU_CLK, the trace port is fixed or variable width and the transmission is at a rate other than CPU_CLK. During asynchronous operation, the input interface of FIFO buffer 246 operates with at the CPU_CLK and the output interface of FIFO buffer 246 operates at the transmission clock of transmission clock ~~generator~~ generation 245. FIFO buffer 246 operates as an asynchronous FIFO buffer between trace packet build 242 and transmission formatter 248. When FIFO buffer 246 is synchronous to CPU_CLK, the trace port is fixed or variable width and the

transmission is at the rate of CPU_CLK. Synchronous operation is similar to asynchronous operation, only the transmission clock and functional clock (CPU_CLK) are the same. A trace packet and port width mismatch requires FIFO buffer 246 between trace packet build 242 and transmission formatter 248 to act as a synchronous FIFO buffer. FIFO buffer 246 is not needed when the trace port is fixed at 10 bits and the transmission is at the rate of CPU_CLK.--

Rewrite the paragraph at page 44, lines 7 to 21 as follows:

--Ring oscillator 305 must be calibrated before the fixed delay option is chosen. Debugger application program 110 establishes the fixed and variable delay values using calibration capabilities built into the local oscillator. This calibration information is then used to choose a fixed delay for normal operation. Calibration involves counting the number of oscillator clocks (OCK) or functional clocks (FCK) that occur within a measurement period defined by a preset number of test clocks (TCK). The measurement period is a window created by a number of test clocks between ~~8* n~~ 8*n where n ranges from 1 to 256, i.e. the clock number ranges from 8 to 2048. Using this measurement requires a minimum TCK frequency of 1MHz. The fixed delay selection and calibration modes are controlled through local oscillator control register LOSC_CNTL 307.--

Rewrite the paragraph at page 44, lines 22 to 29 as follows:

--Local oscillator control register 307 ~~stored~~ stores data decoded as defined in Tables 2 and 3. These define the characteristics of the local oscillator. Local oscillator control register 307 sets the reference clock pre-scaling, the oscillator clock pre-scaling and the operating modes of calibration, test, and normal. This register is part of the trace port register set and

is preferably addresses at base address plus three of the trace port register set.--

Rewrite Table 2 at page 46 as follows:

--

Name	Bit Field	Description
OSCALE[7:0]	15:08	Oscillator Scale ⊟ – LCK divide by 0xFF ⊟ – Divide by 255 0xFE ⊟ – Divide by 255 0xFD ⊟ – Divide by 254 . . . . . . 0x02 ⊟ – Divide by 2 0x01 ⊟ – Divide by 1 0x00 ⊟ – Divide by 256
RSCALE[2:0]	7:5	Reference Scale ⊟ – FCK/TCK divide by: 000 ⊟ – Divide Pre-scaler by 2 001 ⊟ – Divide Pre-scaler by 4 010 ⊟ – Divide Pre-scaler by 8 011 ⊟ – Divide Pre-scaler by 16 100 ⊟ – Divide Pre-scaler by 32 101 ⊟ – Divide Pre-scaler by 64 110 ⊟ – Divide Pre-scaler by 128 111 ⊟ – Divide Pre-scaler by 256
FIXED[1:0]	4:3	Fixed Delay Select 00 ⊟ – 1 lumped delay 01 ⊟ – 2 lumped delays 10 ⊟ – 3 lumped delays 11 ⊟ – 4 lumped delays

Table 2--

Rewrite the paragraph at page 49, lines 6 to 19 as follows:

--Clock scaling 301 includes a 3-bit ripple pre-scaler 413 on the input of the reference clock channel and a 3-bit ripple pre-

scaler 425 on the input of the oscillator clock channel. Reference channel 3-bit pre-scaler 413 is followed by 8-bit ripple counter 415. Oscillator channel 3-bit pre-scaler 425 is followed by 8-bit ripple counter 427. The respective 3-bit pre-scalers 413 and 425 reduce the input clock rate to a frequency that is easily manageable. The length of the pre-scalers is chosen to support test clock TCK reference frequencies as low as 1 MHZ. Oscillator 3-bit pre-scaler 425 creates ripple counter clock and load functions roughly four input clocks apart. In normal mode 000 selected by ~~he~~ the OMODE field, pre-enable decode 423 clears all pre-scaler stages (413, 415, 425, 427, 429) to zero.--

Rewrite the paragraphs at page 52, line 7 to page 53, line 4 as follows:

--Functional clock FCK Measurement:

000 ⊟ = Initialize the local oscillator

100 ⊟ = Initialize the local oscillator

101 ⊟ = Measure functional clock FCK

Read control register until 8 duplicate values other than zero to ascertain completion

Oscillator clock OCK Measurement: From fastest to slowest

000 ⊟ = Initialize the local oscillator

100 ⊟ = Initialize the pre-scalers

111 ⊟ = Measure oscillator clock OCK with variable delay 0

Read control register until 8 duplicate values other than zero to ascertain completion

000 ⊟ = Initialize the local oscillator

100 ⊟ = Initialize the pre-scalers

111 ⊟ = Measure oscillator clock OCK with variable delay 1

Read control register until 8 duplicate values other than zero to ascertain completion

.....

000 ~~⊟~~ _-_ Initialize the local oscillator

100 ~~⊟~~ _-_ Initialize the pre-scalers

111 ~~⊟~~ _-_ Measure oscillator clock OCK with variable delay n--

Rewrite the paragraph at page 66, line 5 to page 67, line 12as follows:

--Figure 13 illustrates the various means for loading the clock control data into local oscillator control register 307. Local oscillator control register 307 may be memory mapped into the memory space of CPU core 201 and thus accessible via CPU bus ~~901~~ 10. Alternatively, local oscillator clock control register 307 may be written into via indirect access register ~~921~~ 21. The data is written to indirect access register ~~921~~ 21 from CPU bus ~~910~~ 10. Indirect access register ~~921~~ 21 passes the data to local oscillator control register 307 via private bus ~~923~~ 23. It is typical for indirect access register ~~921~~ 21 to receive control data together with the data for local oscillator control register 307. This may include routing data so that indirect access register ~~921~~ 21 can service plural indirectly accessed registers. Note that because the real time data exchange process can control all systems available to CPU core 201, these data writes can be controlled external to target system 140 via real-time data exchange export 241, which is preferably bi-directional. As a final alternative, local oscillator control register 307 may be loaded via a JTAG serial scan system including scan-in line ~~931~~ 31 and scan-out line ~~933~~ 33. Any particular target system may use one, two or all these alternatives. In the preferred embodiment, oscillator 307 is loaded via indirect access register ~~921~~ 21 which also provides indirect access to other control registers of the debug system.--